An Asymmetric Fingerprinting Scheme based on Tardos Codes

Ana Charpentier[(a)], Caroline Fontaine[(b)], Teddy Furon[(a)], Ingemar Cox[(c)]

[(a)]INRIA-Rennes research center, Campus de Beaulieu, Rennes, France*
[(b)]CNRS/Lab-STICC/CID, Télécom Bretagne/ITI, Brest, France
[(c)]University College London, Dpt. of Computer Science, London, United Kingdom

Abstract. Asymmetric fingerprinting protocols are designed to prevent an untrustworthy Provider incriminating an innocent Buyer. These protocols enable the Buyer to generate their own fingerprint by themself, and ensure that the Provider never has access to the Buyer's copy of the Work. Until recently, such protocols were not practical because the collusion-resistant codes they rely on were too long. However, the advent of Tardos codes means that the probabilistic collusion-resistant codes are now sufficiently short that asymmetric fingerprint codes should, in theory, be practical.
Unfortunately, previous asymmetric fingerprinting protocols cannot be directly applied to Tardos codes, because generation of the Tardos codes depends on a secret vector that is only known to the Provider. This knowledge allows an untrustworthy Provider to attack traditional asymmetric fingerprinting protocols. We describe this attack, and then propose a new asymmetric fingerprinting protocol, specifically designed for Tardos codes.

1 Introduction

This paper considers a problem arising in the fingerprinting of digital content. In this context, a fingerprint is a binary code that is inserted into a Work for the purpose of protecting it from unauthorized use, or, more precisely, for the purpose of identifying individuals responsible for its unauthorized use. In such a scenario, it is assumed that two or more users may collude in order to try to hide their identities. Under the *marking assumption* [2], colluders cannot alter those bits of the code that are identical for all colluders. However, where bits differ across colluders, these bits may be assigned arbitrary values. A key problem is resistance to collusion, i.e. if c users create a pirated copy of the Work, its tampered fingerprint (i) should not implicate innocent users, and (ii) should identify at least one of the colluders.

This problem has received considerable attention since Boneh and Shaw [2] discussed it. They introduced the concept of a c-secure code such that the probability of framing an innocent user is lower than ϵ. Unfortunately, the length of

* Supported by National Project MEDIEVALS ANR-07-AM-005.

their codes, $O(c^4 \log(\frac{n}{\epsilon}) \log(\frac{1}{\epsilon}))$ where n is the number of users, was too long to be practical. Following Boneh and Shaw's paper, there has been considerable effort to design shorter codes. In 2003, Tardos [19] proposed an efficient code construction that, for the first time, reduced the code length to the theoretical lower bound, $O(c^2 \log(\frac{n}{\epsilon}))$, thereby making such codes practical. Tardos codes are currently the state-of-the-art for collusion-resistant fingerprinting.

Contemporaneously, some papers considered the scenario where the Provider is untrustworthy. Given knowledge of a Buyer's fingerprint, the Provider creates a pirated copy of a Work, implicating the innocent Buyer. To prevent this, Pfitzman and Schunter [16] first introduced the concept of asymmetric fingerprinting in which the Provider does not need to know the Buyer's fingerprint. The Buyer first commits to a secret (the fingerprint) that only he/she knows. The Buyer and Provider then follow a protocol which results in the Buyer receiving a copy of the Work with his/her secret fingerprint (and some additional information coming from the Provider) embedded within it. The Provider does not learn the Buyer's secret, and cannot therefore create a forgery. Unfortunately, the early implementations of this concept were not practical due to the very long length of the collusion resistant codes. The advent of Tardos codes has reduced the length of the collusion resistant codes to a practical size. However, generation of these codes depends on a probability distribution based on a secret vector that is only known to the Provider. This knowledge is sufficient for the Provider to circumvent traditional asymmetric fingerprinting protocols.

In the next Section, we briefly summarize the design of Tardos codes. We then describe how an untrustworthy Provider, with knowledge of the secret vector needed to generate the Tardos codes, can false accuse an innocent Buyer. Section 3 then describes a new asymmetric fingerprinting protocol specific to the use of Tardos codes, that prevents both the Buyer and the Provider from cheating. Practical aspects of the fingerprints embedding and accusation are discussed in Section 4, while security and efficiency of the whole scheme are discussed in Section 6.

2 Untrustworthy Provider with the Tardos code

For readers unfamiliar with Tardos codes, we now provide a brief introduction. Further details can be found in [18].

2.1 Introduction to Tardos codes

Let n denote the number of buyers, and m the length of the collusion-resistant codes. The fingerprints can then be arranged as a binary $n \times m$ matrix $\mathbf{X}$, where Buyer j's binary fingerprint is the jth row of the matrix, i.e. $\mathbf{X}_j = (X_{j1}, X_{j2}, \ldots, X_{jm})$.

To generate this matrix, m real numbers $p_i \in [t, 1-t]$ are generated, each of them being randomly and independently drawn according to the probability density function $f : [t, 1-t] \to \mathbb{R}^+$ with $f(z) = \kappa(t)(z(1-z))^{-1/2}$ and $\kappa(t)^{-1} =$

$\int_t^{1-t}(z(1-z))^{-1/2}dz$. The parameter $t \ll 1$ is referred to as the cutoff whose value is around $1/300c$. The resulting vector, $\mathbf{p} = (p_1, \ldots, p_m)$ is a secret only known by the Provider. Each element of the matrix $\mathbf{X}$ is then independently randomly drawn, such that the probability that the element X_{ji} is set to symbol '1' is $\mathbb{P}(X_{ji} = 1) = p_i$. The collusion-resistant fingerprint, $\mathbf{X}_j$, is then embedded into Buyer j's copy of the Work. This embedding can be accomplished by a variety of watermarking techniques.

When an unauthorized copy is found, a binary sequence, $\mathbf{Y}$, is extracted from the copy thanks to the watermark decoder. Due to collusion and possible distortions such as transcoding, this binary sequence is unlikely to exactly match one of the fingerprints in the matrix $\mathbf{X}$. To determine if Buyer j is involved in the creation of the unauthorized copy, a score, referred to as an accusation score, S_j is computed. If this score is greater than a given threshold Z, then Buyer j is considered to have colluded. The value of the threshold Z theoretically guarantees that the probability of accusing an innocent person is below a significance level, ϵ.

The scores are computed according to an accusation function g, reflecting the impact of the correlation between the fingerprint $\mathbf{X}_j$, associated with Buyer j, and the decoded sequence $\mathbf{Y}$:

$$S_j = G(\mathbf{Y}, \mathbf{X}_j, \mathbf{p}) = \sum_{i=1}^{m} g(Y_i, X_{ji}, p_i). \tag{1}$$

In the usual symmetric codes [18], the function g is constrained (for example, for an innocent person, the expectation of the score is zero and its variance is m), giving $g(1,1,p) = g(0,0,1-p) = -g(0,1,p) = -g(1,0,1-p) = \sqrt{\frac{1-p}{p}}$.

2.2 Untrustworthy content provider

We now consider the case where the Provider is no longer trusted, and wishes to frame Buyer j. There are a number of scenarios, depending on the knowledge available to the Provider. We briefly outline these and discuss our specific scenario in detail.

The Provider knows the Buyer's fingerprint and how to embed the corresponding watermark. This scenario provides no protection to the Buyer. The Provider can simply watermark a Work with the fingerprint of Buyer j, place the Work in an incriminating location and then accuse Buyer j.

The Provider knows the Buyer's fingerprint. In this scenario the Provider does not have the ability to watermark a Work. Instead, upon a Provider's request, a trusted Technology Provider embeds the fingerprint into a Work and sends the fingerprinted Work to the Buyer. We emphasize that the Technology Provider is trusted, and as such, the Provider cannot embed the same fingerprint

into a Work and have it delivered to two different users, one of which is colluding with the Provider to frame the other user. If the Technology Provider were not trusted, we would be back to the previous scenario.

All the Provider needs is fingerprinted copies from $c \geq 3$ fake users or colluders. There is nothing special about the particular fingerprints. For a given Buyer j, whom the Provider wishes to frame, the Provider knows where the elements of the Buyer's fingerprint $X_{ji} = 1$. This happens with probability p_i. At least one of the accomplices has the same symbol as the Buyer with a probability of $1 - (1 - p_i)^c$. Therefore, given that the Provider knows the Buyer's fingerprint, $\mathbf{X}_j$, the accomplices can forge a sequence very similar to the fingerprint of Buyer j. More specifically, if $Y_i = X_{ji}$ whenever the marking assumption allows it, then the forgery is such that, in expectation, the score of Buyer j becomes:

$$\begin{aligned} S_j &= m \int_t^{1-t} f(p) \left[p(1-(1-p)^c) g(1,1,p) + (1-p)(1-p^c) g(0,0,p) \right. \\ &\quad \left. + \; p(1-p)^c g(0,1,p) + (1-p) p^c g(1,0,p) \right] dp \\ &= 2m\kappa(t) \left((1-2t) - 2\frac{(1-t)^{c+1} - t^{c+1}}{c+1} \right) \approx 2m\kappa(t)(1 - \frac{2}{c+1}) \end{aligned} \tag{2}$$

In comparison, the colluders have scores equalling $2m\kappa(t)c^{-1}$ in expectation. This means that with only $c = 3$ accomplices, the score of Buyer j is bigger than the ones of the colluders, which are bigger than Z if the code is long enough to face a collusion of size 3 (depending on the parameters (n, ϵ)). The Provider sends $(\mathbf{X}_j, \mathbf{Y}, \mathbf{p}, Z)$ to the Judge as an evidence to accuse Buyer j. This attack is just an example, there certainly exists a better way to frame an innocent.

The Provider knows the bias vector p. The previous two scenarios demonstrate that the Provider must not know the fingerprints of the Buyers, if the Buyers are to be protected. This is well known in the literature of asymmetric fingerprinting. However, another threat occurs when dealing with Tardos codes. In this scenario, the Provider has no knowledge of the Buyer's fingerprint, nor the underlying watermark method. We therefore assume that the Provider cannot forge an unauthorized copy, either on his/her own or with accomplices. On receipt of a pirated copy, the sequence is extracted by the trusted Technology Provider. Given the extracted sequence $\mathbf{Y}$, the scores of all Buyers are computed using Equation (1). It is here that the Provider can lie, since the probabilities in $\mathbf{p}$ are only known by the Provider.

Specifically, an untrustworthy Provider can create a fake vector of probabilities $\hat{\mathbf{p}}$ that implicates Buyer j. However, the distribution $f(p)$ is publicly known, so the question becomes how to generate a $\hat{\mathbf{p}}$ that (i) implicates Buyer j, and (ii) has an arbitrarily high probability of been drawn from the distribution $f(p)$?

The following method shows that it is simple to do so. However, we do not claim that this attack is unique or optimal. Let us focus on a column where $p_i = p$ and $Y_i = X_{j,i}$. The true summand in Equation (1) is $g(1,1,p)$ or $g(0,0,p)$ (with equal probability). Suppose that the content provider replaces the secret

value p by a fake secret $\hat{p}$ which is drawn independently according to f. On average, this summand takes the new value:

$$\Delta(t) = \int_t^{1-t} f(\hat{p}) \frac{g(1,1,\hat{p}) + g(0,0,\hat{p})}{2} d\hat{p} = \kappa(t) \ln \frac{1-t}{t}.$$

For a cutoff $t = 1/900$ (recommended by G. Tardos to fight against 3 colluders), $\kappa(t) \approx \pi^{-1}$ and the numerical value is surprisingly high: $\Delta(1/900) \approx 2.16$. Suppose now that the content provider applies the same strategy on an index i where $Y_i \neq X_{j,i}$. Then the expectation is the opposite. However, in a Tardos code, even for an innocent Buyer j, the proportion α of indices where symbols Y_i and $X_{j,i}$ agree is above $1/2$ for common collusion strategies. For instance, with an interleaving collusion attack [18], $\alpha = 3/4$ whatever the collusion size c.

Based on this fact, we propose the following attack. The Provider computes the score for all Buyers, which on average equals 0 for innocent Buyers and $2m\kappa(t)c^{-1}$ for the colluders [18]. The Provider initializes $\hat{\mathbf{p}} = \mathbf{p}$. Then, he/she randomly selects a column i and randomly draws a fake secret $\hat{p}_i \sim f$. He/She re-computes the score of Buyer j with this fake secret and iterates selecting a different column until S_j is above the threshold Z. On average, $m(c\kappa(t)^{-1}\Delta(t)(\alpha - 1/2))^{-1}$ secret values p_i need to be changed in this way, e.g. only 20% of the code length if the copy has been made using an interleaving attack.

Figure 1 illustrates this attack for the case where the code length is $m = 1000$ and the number of colluders is $c = 3$. The solid coloured lines depict the accusation scores of 10 randomly selected innocent buyers. We observe that after 20 to 30% of the elements of $\mathbf{p}$ have been altered, the accusation scores of the innocent Buyers exceed the *original* scores of the colluders. In fact, the colluders' accusation scores also increase. However, we are not concerned by the highest score, but rather by the fact that the Provider is able to exhibit a couple $(\hat{\mathbf{p}}, \mathbf{X}_j)$ such that $S_j > Z$. Thus, it is sufficient to raise the score of the innocent Buyer, even if this raises all other Buyers' scores as well.

Randomly selecting some p_i's (independently from $\mathbf{X}_j$ and $\mathbf{Y}$) and re-drawing them according to the same law ensures that $\hat{p}_i \sim f$, $\forall i$. Therefore, the Judge observing $\hat{\mathbf{p}}$ cannot distinguish the forgery. For this reason, the Judge might request to see the matrix $\mathbf{X}$ to statistically test whether the elements of $\mathbf{X}$ are drawn from the distribution $\hat{\mathbf{p}}$. In this case, the Provider can give a fake matrix $\hat{\mathbf{X}}$ where the columns whose p_i have been modified are re-drawn such that $\mathbb{P}(X_{ki} = 1) = \hat{p}_i$, $\forall k \neq j$. The only way to prevent this deception would be for the Judge to randomly asked an innocent Buyer $k \neq j$ for his copy in order to verify the authenticity of $\hat{\mathbf{X}}$. This latter step seems somewhat odd. We arrive at the strange situation where the Judge has to contact innocent buyers when Buyer j is accused.

3 An asymmetric Tardos code construction

The previous section underlines the difficulty of constructing an asymmetric fingerprinting protocol using Tardos codes. The constraints are:

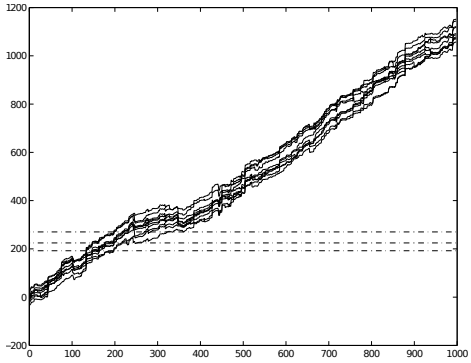

Fig. 1. Accusation score as a function of the number of changed elements of the vector $\mathbf{p}$ for the case where $m = 1000$ and $c = 3$. The solid coloured lines show how the accusation scores of 10 randomly selected innocent buyers increases. The dotted horizontal lines show the original scores for the colluders before the modification.

- The Provider should not know the fingerprints.
- The Provider should not change the secret $\mathbf{p}$ used for the code construction during the accusation score computation.
- The Buyer should know neither the secret $\mathbf{p}$ nor the fingerprint of any other user.
- His fingerprint must be drawn according to the statistical distribution induced by $\mathbf{p}$.
- The Buyer should not be able to modify his fingerprint.

These constraints prevent the application of previous asymmetric fingerprinting schemes to a Tardos code. This section proposes a solution to this problem, which consists of two phases: the generation of the fingerprint and the disclosure of a halfword. Both phases rely on a primitive which we present first.

3.1 Pick a card, any card!

What we need is a scheme that enables a receiver $\mathbf{R}$ to pick k elements at random in a list of N elements provided by a sender $\mathbf{S}$, in such a way that:

1. $\mathbf{R}$ gets elements that belong to the list;
2. $\mathbf{R}$ does not get any information on the elements he did not pick;
3. $\mathbf{S}$ does not know which elements have been picked.

Functionally speaking, this is precisely what is called *Oblivious Transfer* by cryptographers. A k-out-of-N Oblivious Transfer protocol is denoted by OT_k^N. In the literature we can find OT_1^2, OT_1^N and OT_k^N protocols. When $k \geq 1$, if the k elements are picked one-by-one adaptively, we speak of *adaptive OT protocols*, denoted by $OT_{k\times 1}^N$; if they are picked simultaneously, we speak of *non-adaptive OT protocols*, simply denoted OT_k^N .

Technically speaking, the oblivious transfer problem has been independently tackled by two communities. First, Cryptographers have been working on it since 1981. We will refer to this quite long and mature framework as "traditional" OT. Second, in 2001 other researchers proposed a different approach based on *Commutative Encryption* and *Two-lock Cryptosystems*. Both are considered and discussed in Sec. 4, according to their respective advantages. We provide more details on the use of OT protocols based on Commutative Encryption or Two-lock crypto-systems, as they are less known but particularly interesting in our case.

3.2 Phase 1: Generation of the fingerprint

Fingerprint generation consists of two steps. During Step 1, the Provider generates lists from the secret $\mathbf{p}$, and commits them in order to avoid any *a posteriori* cheating. During Step 2, the Buyer picks elements in the lists to generate his own fingerprint. This step is addressed by oblivious transfer protocols.

Step 1. We use the commutative encryption protocol m times to generate the fingerprint of the j-th Buyer $\mathbf{X}_j = (X_{j,1}, \ldots, X_{j,m})$. $\mathbf{S}$ is the Provider, and $\mathbf{R}$ is Buyer j. The Provider generates a secret vector $\mathbf{p}$ for a Tardos code. Each p_i is quantized such that $p_i = L_i/N$ with $L_i \in [N-1]$.

For a given index i, the objects are the concatenation of a binary symbol and a text string. There are only two versions of an object in list $\mathcal{C}_i$. For L_i objects, $O_{k,i} = (1 \| \mathtt{ref}_{1,i})$, and $O_{k,i} = (0 \| \mathtt{ref}_{0,i})$ for the $N - L_i$ remaining ones. The use of the text strings $\{\mathtt{ref}_{X,i}\}$ depends on the content distribution mode as detailed in Sec. 4.3. The object $O_{k,i}$ is committed with key $K_{k,i}$ and stored in the list $\mathcal{C}_i = \{C_{k,i}\}_{k=1}^N$. There are thus as many different lists $\mathcal{C}_i$ as the length m of the fingerprint. These lists are the same for all buyers, and are published in a public Write Once Read Many (WORM) directory [15] whose access is granted to all users. As the name, nobody can modify or erase what is initially written in a WORM directory, but anyone can read from it.

$$
\begin{array}{lcll}
p_1 & \overset{Quantize}{\longrightarrow} & (0\|\mathtt{ref}_{0,1}, 1\|\mathtt{ref}_{1,1}, \ldots, 1\|\mathtt{ref}_{1,1}) & \overset{Commit}{\longrightarrow} \quad \mathcal{C}_1 = (C_{1,1}, \ldots, C_{N,1}) \\
p_2 & \longrightarrow & (0\|\mathtt{ref}_{0,2}, 0\|\mathtt{ref}_{0,2}, \ldots, 0\|\mathtt{ref}_{0,2}) & \longrightarrow \quad \mathcal{C}_2 = (C_{1,2}, \ldots, C_{N,2}) \\
 & & \vdots & \\
p_m & \longrightarrow & (1\|\mathtt{ref}_{1,m}, 0\|\mathtt{ref}_{0,m}, \ldots, 1\|\mathtt{ref}_{1,m}) & \longrightarrow \quad \mathcal{C}_m = (C_{1,m}, \ldots, C_{N,m})
\end{array}
$$

Fig. 2. The lists $\mathcal{C}_i = \{C_{k,i}\}_{k=1}^N$ are stored in a WORM

Step 2. If we use a traditional Oblivious Transfer protocol, the Buyer and Provider run it to get the corresponding key $K_{\mathsf{ind}(j,i),i}$: the Provider proposes the list of the keys $\{\pi_j(k) \| K_{\pi_j(k),i}\}$ and the Buyer picks one with an OT_1^N. This

key allows him to open one of the commitments $C_{\pi_j(k),i}$. Provider and Buyer will have to keep in a log file some elements of the exchange in order to run the Phase 2. It is specific to the *OT* protocol and we have not studied this problem in detail.

Let us now describe how to solve the problem with a Commutative Encryption scheme. Contrary to the $\mathcal{C}$-lists, the $\mathcal{D}$-lists are made specific to a given Buyer j. The Provider picks a secret key S_j and a permutation $\pi_j(.)$ over $[N]$. The Buyer is given a list $\mathcal{D}_{j,i} = \{D_{j,i,k} = \mathbf{CE}(S_j, (\pi_j(k) \| K_{\pi_j(k),i}))\}_{k=1}^{N}$. Therefore, the lists $\{\mathcal{C}_i\}_{i=1}^{m}$ are common for all users, whereas the lists $\{\mathcal{D}_{j,i}\}_{i=1}^{m}$ are specific to Buyer j. We have introduced here a slight change with respect to protocol 4.1, i.e. the permutation π_j whose role is explained below. Buyer j chooses one object in the list, say the $k(j,i)$-th object. He/she sends the corresponding ciphertext $U_{k(j,i),i} = \mathbf{CE}(R_{j,i}, D_{j,i,k(j,i)})$ decrypted by the provider with S_j and sent back to the Buyer who, at the end, gets the index $\mathsf{ind}(j,i) = \pi_j(k(j,i))$ and the key $K_{\mathsf{ind}(j,i),i}$, which grants him/her the access to the object $O_{\mathsf{ind}(j,i),i}$, stored in encrypted form in the WORM. It contains the symbol $b_{\mathsf{ind}(j,i),i}$. This becomes the value of the i-th bit of his/her fingerprint, $X_{j,i} = b_{\mathsf{ind}(i,j),i}$, which equals '1' with probability p_i. The provider keeps in a log file the values of S_j and $U_{k(j,i),i}$, and the user keeps $R_{j,i}$ in his/her records.

	P		B
1.	**Commit** $C_{\pi_j(k),i} = \mathbf{Com}(K_{\pi_j(k),i}, O_{\pi_j(k),i})$		
	Only one time for all users and all bits		
2.	$D_{j,i,k} = \mathbf{CE}(S_j, (\pi_j(k) \| K_{\pi_j(k),i}))$		
3.		$\longleftarrow$	$U_{k(j,i),i} = \mathbf{CE}(R_{j,i}, D_{j,i,k(j,i)})$
4.	$V = \mathbf{CE}^{-1}(S_j, U_{k(j,i),i})$	$\longrightarrow$	
5.			$\pi_j(k) \| K_{\pi_j(k),i} = \mathbf{CE}^{-1}(R_{j,i}, V)$
6.			$O_{\pi_j(k),i} = \mathbf{E}^{-1}(K_{\pi_j(k),i}, C_{\pi_j(k),i})$

Fig. 3. Generation of a fingerprint bit using the Commutative Encryption Scheme.

3.3 Phase 2: Disclosure of the halfword

The accusation process detailed in Sec. 4.4 allows the Provider to list a set of suspected users to be forwarded to the judge for verification. After phase 1 is completed, the Provider orders Buyer j to reveal $m_h < m$ bits of his fingerprint. These disclosed symbols compose the so-called halfword [16]. The following facts must be enforced: Buyer j does not know which bits of his/her fingerprint are disclosed even if the Provider asks for the same bit indices to all the users. The Provider discloses m_h bits of the fingerprints without revealing any knowledge about the others. Of course, Buyer j refuses to follow the protocol for more than m_h objects.

Commutative Encryption. Again, we propose to use the double-blind random selection protocol of Sec. 3.1. Now, Buyer j plays the role of $\mathbf{S}$, and the Provider the role of $\mathbf{R}$, $N = m$, and object $O_i = (R_{i,j} \| \texttt{alea}_{i,j})$. These items are the m secret keys selected by Buyer j during phase 1 (Sec. 3.2) concatenated with random strings $\texttt{alea}_{i,j}$ to be created by Buyer j. This alea finds its use during the personalization of the content (see Sec. 4.3). Following the protocol, the Provider selects m_h such object. The decryption of message $U_{k(i,j),j}$ received during phase 1 thanks to the disclosure of the key $R_{i,j}$ yields $D_{i,j,k(i,j)}$ which in turn is decrypted with key S_j, provides the index of the selected object, otherwise the protocol stops. This prevents a colluder from denying the symbol of his fingerprint and from copying the symbol of an accomplice. At the end, the Provider learns which item was picked by Buyer j at index i. Therefore, he/she ends up with m_h couples $(X_{j,i}, \texttt{alea}_{k(i,j),i})$ associated to a given Buyer j.

Generic Oblivious Transfer protocols. At phase 2, any $OT^N_{k\times 1}$ can be used to allow the Provider to get m_h objects from the list of the $O_i = (R_{i,j} \| \texttt{alea}_{i,j})$ owned by the Buyer. The problem is if another OT scheme was used at the precedent step, there is no such things as the $R_{i,j}$ values. In order to prevent the Buyer from denying the symbol of his fingerprint, the $R_{i,j}$ values have to be replaced by a number which was part of the exchange during the generation of the fingerprint. This element is specific to the OT protocol.

4 Implementation details

The previous section has detailed the core of our scheme which is the construction of the codewords based on oblivious transfer. This section deals with the details of this primitive and the remaining elements, namely the watermarking of video content, the distribution and the accusation process.

4.1 Details of the oblivious transfer protocol

This protocol can be implemented by two approaches, 'classical' Oblivious transfer and Commutative encryption, which have been studied with different security models. Both are interesting for us, and we will now summarize them and discuss their usefulness

Traditional Oblivious Transfer protocols. Oblivious Transfer Protocols have been introduced by cryptographers in [17] and led to a huge number of papers in the cryptographic community, *e.g.* [13, 5, 10]. These protocols are studied in the same framework as multi-party computation. Their security is studied under different models below, listed from the weakest to the strongest: honest-but-curious model (where no one cheats during the protocol execution), half simulation (introduced by [14], cheating sender or cheating receiver studied separately; local security study), full simulation (introduced in [3], studying cheating sender and receiver

globally; global security study). In addition, the UC (Universally Composable) model has been introduced in [4] to study the behavior and security of protocols that are based on concurrent and composable cryptographic primitives.

Oblivious Transfer based on Commutative Encryption. An encryption primitive **CE** is said to be a *Commutative Encryption* if for any two keys k_R and k_S and any plaintext m, we have (usual definition in the literature)

$$\mathbf{CE}(k_R, \mathbf{CE}(k_S, m)) = \mathbf{CE}(k_S, \mathbf{CE}(k_R, m)). \tag{3}$$

Based on such a primitive, a *Commutative Encryption Scheme (CES)* can be defined as follows [1].

1. Let m_1, m_2, ...,m_N be the N inputs of the Sender **S**. **S** chooses N secret keys K_1, K_2, ..., K_N for a symmetric cryptosystem **E** (*e.g.* AES, DES) and a key k_S for the commutative encryption primitive **CE**. **S** provides

$$\begin{aligned} C_1 &= \mathbf{E}(K_1, m_1)\ , \quad & D_1 &= \mathbf{CE}(k_S, K_1) \\ C_2 &= \mathbf{E}(K_2, m_2)\ , \quad & D_2 &= \mathbf{CE}(k_S, K_2) \\ &\ldots \quad & &\ldots \\ C_N &= \mathbf{E}(K_N, m_N)\ , \quad & D_N &= \mathbf{CE}(k_S, K_N) \end{aligned}$$

   Note that the couples $\langle C_j,\ D_j \rangle$ can be publicly accessed.

2. Now, let us assume that the receiver **R** wants to pick the i-th element of the list. **R** loads $\langle C_i,\ D_i \rangle$ and chooses a secret key k_R for **CE**. He encrypts D_i with it and sends the result $U = \mathbf{CE}(k_R, D_i)$ to **S**.
3. **S** decrypts U with S and sends $W = \mathbf{CE}^{-1}(k_S, U)$ to **R**. **R** computes K_i, and can get to $m_i = \mathbf{E}^{-1}(K_i, C_i)$.

A *Two-lock Cryptosystem* is a variant that uses two different primitives **CE1** and **CE2** instead of **CE**:

$$\mathbf{CE1}(k_R, \mathbf{CE2}(k_S, m)) = \mathbf{CE2}(k_S, \mathbf{CE1}(k_R, m)). \tag{4}$$

Both approaches are interesting for us, as we will discuss now. First of all, the security of Oblivious Transfer Protocols has been much stronger studied than the one of the Commutative Encryption Schemes. Hence, we will use them each time it is possible, leaning on well known protocols.

But, at some steps of the protocol we prefer to use Commutative Encryption Schemes, as its structure fits really well to our purpose. It is for example the case during fingerprint generation, as we also want the Provider to commit on the lists elements, which correspond to the secret vector Tardos accusation will rely on. This ensures that the same secret vector will be used during the accusation process. Such commitments are easily included in a Commutative Encryption Scheme, it is more difficult in a traditional Oblivious Transfer protocol. In addition, we use some elements exchanged during the course of the protocol in phase 1 (Sec. 3.2) to ensure the correct conduct of the Phase 2 (Sec. 3.3).

Designing the right Commutative Encryption Scheme is not so easy, as the literature does not provide us a scheme that fulfill our requirements. First of all, notice that using a symmetric or asymmetric encryption primitive as **CE**, or in the variant scheme **CE1** and **CE2**, does not matter here, functionally speaking, as encryption and decryption will be performed by the same person. Hence, only security and eventually efficiency may guide our choice. Of course, we would like to use the most secure encryption primitives. The highest security level, *unconditional security* is only reached by the One-Time Pad, and cannot be achieved here because it would require to use a different key for each encryption whereas here the same key k_S is used to encrypt all the keys K_i. Hence, *semantic security* is the best security class we might achieve [9, 20, 7]. Moreover, semantic security is necessary in our case, because we have to encrypt binary symbols and do not want the Receiver to be able to distinguish encrypted 0's from encrypted 1's during both the fingerprint generation or the halfword disclosure steps. This implies the use of a probabilistic encryption scheme. Unfortunately, semantic security has not yet been tackled in the Commutative Encryption literature [1, 11, 21]. Nevertheless, semantic security should be achieved in a near future, making this kind of OT particularly interesting for us.

Concerning the variant called Two-lock Cryptosystem, a few implementations have been proposed: a first one based on the Knapsack problem [21], which has been broken [22], a second one based on the discrete logarithm problem [21], and a third one based on RSA [11]. None of them achieve semantic security at the moment.

4.2 Watermarking

A nowadays trend is the application of fingerprinting to premium video contents. Premium means movies in very high quality available for home cinema shortly after their release in theaters. Personalization of the copies are usually done as follows: Before distribution, the content is divided into sequential blocks (e.g. Group of Pictures of few seconds of a video). Offline, a robust watermarking technique creates two versions of some blocks embedding the symbol '0' and respectively '1'. This is done by the Technology Provider. Quality is very important for premium movies and watermarking under that constraint involves a lot of processing. This motivates this offline preprocessing.

In some scenarios (screeners for jurys, marketing, blu-ray discs, premium downloads), the physical medium storage or bandwitth is so large that both versions of the blocks are encrypted and transmitted to the software client or the device of the Buyers. This latter is trusted and the strings $\{\texttt{ref}_{X,i}\}$ it got from phase 1 are parameters needed to get access to the i-th block watermarked with symbol X.

4.3 Content personalization at the server side

As for Video On Demand where the client is not trusted, personalization of the content is usually made at the server side, which raises an issue since the Provider

doesn't know user fingerprints. There exist Buyer-Seller protocols for embedding a sequence $\mathbf{X}_j$ into a content c_o without disclosing $\mathbf{X}_j$ to the Seller and c_o to the Buyer. They are based on homomorphic encryption scheme and work with some specific implementations of spread spectrum [12] or Quantization Index Modulation watermarking [6]. In other words, not any watermarking technique can be used, and this is not the route we have chosen so far. Due to space limitations, a brief sketch of the adaptation of [6] is presented hereafter.

Let $\mathbf{c}_i^{(0)} = (c_{i,1}^{(0)}, \ldots, c_{i,Q}^{(0)})$ be the Q quantized components (like pixels, DCT coefficients, portion of streams etc) of the i-th content block watermarked with symbol '0' (resp. $\mathbf{c}_i^{(1)}$ with symbol '1'). Denote $\mathbf{d}_i = \mathbf{c}_i^{(1)} - \mathbf{c}_i^{(0)}$. Assume as in [6, Sect. 5], an additive homomorphic and probabilistic encryption $E[.]$ such as the Pallier cryptosystem. Buyer j has a pair of public/private keys (pk_j, sk_j) and sends $(E_{pk_j}[X_{j,1}], \ldots, E_{pk_j}[X_{j,m}])$. The provider sends him/her the ciphers

$$E_{pk_j}[c_{i,\ell}^{(0)}].E_{pk_j}[X_{j,i}]^{d_{i,\ell}}, \forall (i,\ell) \in [m] \times [Q]. \tag{5}$$

Thanks to the homomorphism, Buyer j decrypts this with sk_j into $c_{i,\ell}^{(0)}$ if $X_{j,i} = 0$, $c_{i,\ell}^{(1)}$ if $X_{j,i} = 1$. Since $X_{j,i}$ is constant for the Q components of the i-th block, a lot of bandwidth and computer power will be saved with a composite signal representation as detailed in [6, Sect. 3.2.2].

A crucial step in this kind of Buyer-Seller protocols is to prove to the Provider that what is sent by the Buyer is indeed the encryption of bits, and moreover bits of the Buyer's fingerprint. This usually involves complex zero-knowledge subprotocols [12, 6]. Here, we avoid this complexity by taking advantage of the fact that the Provider already knows some bits of the fingerprint $\mathbf{X}_j$, i.e. those belonging to the halfword (see Sec. 3.3), and the Buyers do not know the indices of these bits. Therefore, in $m_v < m_h$ random indices of the halfword, the Provider asks Buyer j to open his/her commitment. For one such index i_v, Buyer j reveals the random value r_{i_v} of the probabilistic Pallier encryption (with the notation of [6]). The Provider computes $g^{X_{j,i_v}} h^{r_{i_v}} \mod N$ and verifies it equals the i_v-th cipher, which Buyer j pretended to be $E_{pk_j}[X_{j,i}]$.

One drawback of this simple verification scheme is that the Buyer discovers m_v indices of the halfword. This may give rise to more elaborated collusion attacks. For example, Buyer j, as a colluder, could try to enforce $Y_{i_v} \neq X_{j,i_v}$ when attempting to forge a pirated copy. Further discussion of this is beyond the scope of this paper.

This approach may also introduce a threat to the Buyer. An untrustworthy Provider can ask to open the commitments of non-halfword bits in order to disclose bits he/she is not supposed to know. For this reason, the Provider needs to send $\texttt{alea}_{k(i_v,j),i_v}$ as defined in Sec. 3.3 to show Buyer j that his/her verification duly occurs on a halfword bit.

4.4 The accusation procedure

The accusation is straightforward and similar to other fingerprinting protocols. A Scouting Agency is in charge of catching a forgery. The Technology Provider

decodes the watermark and extracts sequence $\mathbf{Y}$ from the pirated content. The Provider computes the halfscores by applying Eq. (1) only on the halfwords. This produces a list of suspects, e.g. those users whose score is above a threshold, or those users with the highest scores.

Of course, this list cannot be trusted, since the Provider may be untrustworthy. The list is therefore sent to a third party, referred to as the Judge, who first verifies the computation of the halfscores. If different values are found, the Provider is black-listed. Otherwise, the Judge computes the scores of the full fingerprint.

To do so, the Judge needs the secret $\mathbf{p}$: he/she asks the Provider for the keys $\{K_{k,i}\}$, $\forall (k,i) \in [N] \times [m]$ and thereby obtains from the WORM all the objects $\{O_{k,i}\}$, and the true values of $(p_1, \ldots, p_m)$. The Judge must also request suspected Buyer j for the keys $R_{j,i}$ in order to decrypt the messages $U_{k(j,i),i}$ in $D_{i,j,k(i,j)}$ which reveal which object Buyer j picked during the i-th round of Sec. 3.2 and whence $X_{j,i}$. Finally, the Judge accuses the user whose score over the full length fingerprint is above a given threshold (related to a probability of false alarm).

5 Discussion

5.1 Security

Suppose first that the Provider is honest and denote by c the collusion size. A reliable tracing capability on the halfwords is needed to avoid false alarms. Therefore, as proven by G. Tardos, $m_h = O(c^2 \log n \epsilon^{-1})$, where ϵ is the probability of suspecting some innocent Buyers. Moreover, successful collusions are avoided if there are secret values such that $p_i < c^{-1}$ or $p_i > 1 - c^{-1}$(see [8]). Therefore, N should be sufficiently big, around a hundred, to resist against collusion of size of some tens. During the generation of the fingerprint in Sec. 3.2, permutation $\pi_j(.)$ makes sure that Buyer j randomly picks up a bit '1' with probability $p_i = L_i/N$ as needed in the Tardos code. In particular, a colluder cannot benefit from the discoveries made by his accomplices.

We now analyze why colluders would cheat during the watermarking of their version of the Work described in Sec. 4.3. By comparing their fingerprints, they see indices where they all have the same symbols, be it '0' or '1'. As explained in the introduction, they won't be able to alter those bits in the tampered fingerprint except if they cheat during the watermarking: If their fingerprint bits at index i all equal '1', one of them must pretend he/she has a '0' in this position. If they succeed to do so for all these positions, they will able to forge a pirated copy with a null fingerprint for instance.

How many times do the colluders need to cheat? With probability p_i^c (resp. $(1-p_i)^c$), they all have bit '1' (resp. '0') at index i. Thus, there are on average $m_c(c) = m \int_t^{1-t} (p^c + (1-p)^c) f(p) dp$ such indices. The Provider asks for a bit verification with probability m_v/m_h. The probability of a successful attack for a collusion of size c is therefore $(1 - m_v/m_h)^{m_c(c)}$. Our numerical simulations

(see figure 4 (a)) show that m_v shouldn't be more than 50 bits for typical code length and collusion size below a hundred. Thus, m_v is well below m_h.

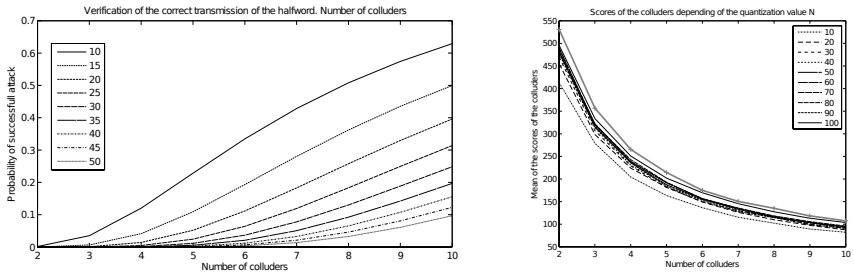

Fig. 4. (a) m_v goes from 10 to 50 by 5, $m = 3000$ and $m_h = 1500$. (b) N goes from 10 to 100 and $m = 1500$. The gray curve with crosses is for the unquantified Tardos code.

Suppose now that the Provider is dishonest. The fact that the m lists $\mathcal{C}_i$, $\forall i \in [m]$ are public and not modifiable prevents the Provider from altering them for a specific Buyer in order to frame him/her afterwards. Moreover, it will raise the Judge's suspicion if the empirical distribution of the p_i is not close to the pdf f. Yet, biases can be introduced on the probabilities for the symbols of the colluders' fingerprint only if there is a coalition between them and the untrustworthy Provider. For instance, the Provider can choose a permutation such that by selecting the first item (resp. the last one) in the list $\mathcal{D}_{j,i}$ an accomplice colluder is sure to pick up a symbol '1' (resp. '0'). This ruins the tracing property of the code, but this does not allow the Provider to frame an innocent. First, it is guaranteed that $\mathbf{p}$ used in Eq. (1) is the one which generated the code. Second, the Provider and his accomplices colluders must ignore a significant part of the fingerprints of innocent Buyers. To this end, $m - m_h$ must also be in order of $O(c^2 \log n\epsilon^{-1})$. If this holds, the Judge is able to take a reliable decision while discarding the halfword part of the fingerprint. Consequently, $m \approx 2m_h$, our protocol has doubled the typical code length, which is still in $O(c^2 \log n\epsilon^{-1})$.

5.2 Efficiency

Parameters The parameters of the Tardos code are chosen according to the formulas linking length, number of colluders, and number of users. We have found out that the value m_v doesn't need to be more than 50, see Sec. 4. We consider the value N, the quantization parameter, with the interleaving collusion attack. In the figure 4 (b), we can see that up to a small value of N (around 20), there is no gain of efficiency. The red line shows that the results with the unquantized Tardos parameters remain better.

Complexity The cost of phase 1 is $m \times N$ commitments for the lists that will be stored in the Worm file, and $mn \times (N+4)$ exponentiations for the OT phase.

Regarding the use of a non specific OT, still $m \times N$ commitments, plus the cost of mn 1-out-of-N Oblivious Transfers. This cost depends of course of the chosen protocol, it is in $O(N)$ for a lot of protocols. For Phase 2, the cost is that of an m_h-out-of-m Oblivious transfer. If this OT is performed with the use of a *Commutative Encryption*, the cost is $2m+4m_h$ for the communication, and $4m_h$ rounds, for another OT scheme, the communication is in $O(m)$ and the number of rounds depends of the protocol, it is usually in $O(m_h)$.

6 Conclusion

Tardos codes are currently the state-of-the-art in collusion-resistant fingerprinting. However, the previous asymmetric fingerprint protocols cannot be applied to this particular construction. There are mainly two difficulties. First, the Buyer has to generate his/her secret fingerprint but according to vector $\mathbf{p}$, which is kept secret by the Provider. Second, the secret $\mathbf{p}$ used in the accusation process must be the same as the one which generated the fingerprints.

We have proposed the first asymmetric fingerprinting protocol dedicated to Tardos codes. The construction of the fingerprints and their embedding within pieces of Work do not need a trusted third party. Note, however, that during the accusation stage, a trusted third party is necessary like in any asymmetric fingerprinting scheme we are aware of. Further work is needed to determine if such a third party can be eliminated. In particular, we anticipate that some form of secure multi-party computation can be applied.

We considered two forms of oblivious transfer protocols, the first based on traditional cryptographic techniques and the second based on less well known Commutative Encryption or Two-Lock crypto-systems. These latter techniques are less mature than traditional Oblivious Transfer protocols in terms of security, but offers interesting properties that are convenient to our application. Further work is needed to improve their semantic security, so that their advantages do not come at the cost of decreased security.

7 Acknowledgement

We would like to thank Boris Škorić, and the three anonymous reviewers for their useful comments, which helped to improve the presentation of our results.

References

1. Bao, F., Deng, R., Feng, P.: An efficient and practical Scheme for Privacy Protection in the E-Commerce of Digital Goods. In: ICISC 2000. LNCS, vol. 2015, pp. 162–170. Springer-Verlag (2001)
2. Boneh, D., J.Shaw.: Collusion-secure fingerprinting for digital data. IEEE Trans. Inform. Theory (1998)
3. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. Advances in Cryptology – EUROCRYPT 2007 4515, 573–590 (2007)

4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on. pp. 136–145. IEEE (2002)
5. Chu, C., Tzeng, W.: Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries. In: Public Key Cryptography – PKC 2005. LNCS, vol. 3386, pp. 172–183. Springer-Verlag (2005)
6. Deng, M., Bianchi, T., Piva, A., Preneel, B.: An efficient Buyer-Seller watermarking protocol based on composite signal representation. In: ACM MM&Sec'09. pp. 9–18 (2009)
7. Fontaine, C., Galand, F.: A survey of homomorphic encryption for nonspecialists. EURASIP Journal on Information Security 2007, 15 (2007)
8. Furon, T., Pérez-Freire, L.: Worst case attack against binary probabilistic traitor tracing codes. In: IEEE WIFS 2009. pp. 46–50 (2009)
9. Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge Univ Pr (2004)
10. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Advances in Crypotology – ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer-Verlag (2007)
11. Huang, H., Chang, C.: A new design for efficient t-out-n oblivious transfer scheme (2005)
12. Kuribayashi, M.: On the Implementation of Spread Spectrum Fingerprinting in Asymmetric Cryptographic Protocol. EURASIP Journal on Inf. Security (2010)
13. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Advances in Cryptology – CRYPTO99. LNCS, vol. 1666, pp. 791–791. Springer-Verlag (1999)
14. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. Journal of Cryptology 18(1), 1–35 (2005)
15. Oprea, A., Bowers, K.D.: Authentic Time-Stamps for Archival Storage. In: ESORICS 2009. LNCS, vol. 5789, pp. 136–151. Springer-Verlag (2009)
16. Pfitzmann, B., Schunter, M.: Asymmetric fingerprinting. In: EUROCRYPT 96. LNCS, vol. 1070, pp. 84–95. Springer-Verlag (1996)
17. Rabin, M.: How to exchange secrets by oblivious transfer. Tech. rep., Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981 (1981)
18. Skoric, B., Katzenbeisser, S., Celik, M.: Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes. Designs, Codes and Cryptography 46(2), 137–166 (2008)
19. Tardos, G.: Optimal probabilistic fingerprint codes. In: STOC 2003. pp. 116–125. ACM (2003), `http://www.renyi.hu/~tardos/publications.html`
20. van Tilborg, H.: Encyclopedia of cryptography and security. Springer Verlag (2005)
21. Wu, Q., Zhang, J., Wang, Y.: Practical t-out-n oblivious transfer and its applications. In: Information and Communications Security. LNCS, vol. 2936, pp. 226–237. Springer-Verlag (2003)
22. Zhang, B., Wu, H., Feng, D., Bao, F.: Cryptanalysis of a knapsack based two-lock cryptosystem. In: Applied Cryptography and Network Security. pp. 303–309. Springer-Verlag (2004)